ERDC MSRC/PET TR/00-12

Scientific Visualization of Water Quality in the Chesapeake Bay

by

Robert Stein
Alan M. Shih
M. Pauline Baker
Carl F. Cerco
Mark R. Noel

3 April 2000

07h0012000

Scientific Visualization of Water Quality in the Chesapeake Bay

Robert Stein[1], Alan M. Shih[2], M. Pauline Baker[3]
National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign

Carl F. Cerco[4], Mark R. Noel[5]
U. S. Army Engineer Research and Development Center

Abstract

This paper describes our experience in designing and building a tool for visualizing the results of the CE-QUAL-ICM Three-Dimensional Eutrophication Model, as applied to water quality in the Chesapeake Bay. This model outputs a highly multidimensional dataset over very many timesteps – outstripping the capabilities of the visualization tools available to the research team. As part of the Army Engineer Research and Development Center (ERDC) Programming Environment and Training (PET) project, a special visualization tool was developed. This paper includes discussions on how the simulation data are handled efficiently, as well as how the issues of usability, flexibility and collaboration are addressed.

Introduction

The Chesapeake Bay is the largest and most productive estuary in the United States. With a surrounding population of about 15 million people, and a valuable place in the fishing industry, the Chesapeake Bay is a very important natural resource to the region. Population growth and the development of agriculture and industry in the area surrounding the bay have had a significant impact on this ecosystem.

Scientists at the U. S. Army Engineer Research Development Center (ERDC) and the Environmental Protection Agency (EPA) have developed a computational model to help them understand the effects of pollutants and contaminants on the water quality of the Chesapeake Bay. This computational system allows scientists to validate and predict the water quality in the bay through dozens of constituents produced by the simulation.

Comparing model simulations with existing field observations and examining model predictions of future environmental scenarios, not only require statistical measurements, but more importantly, require visualization. Visualization of this model has been through many derivations and software packages, both commercial and customized.

Available commercial visualization tools are sometimes tedious to use for handling both the unique data format used by the computational model and the particular needs of the researchers. Therefore, a custom visualization tool was developed to accurately and efficiently deal with the data from this model, and to aid in analysis of the data by applying a set of advanced visualization algorithms. Sample results are presented in this paper.

Before the development of this tool scientists used standard X-Y line graphs and simple 2-dimensional slices to visualize the model prediction. S.C.I.R.T., the Site Characterization Interactive Research Toolkit, [6] was the first visualization tool which included 3-dimensional comparisons of model predictions and observed data and 3-dimensional views of flow/flux vector predictions.

Requirements

In developing a user-friendly visualization package tailored for the Chesapeake Bay water quality simulation project, we found that the user had several needs. First it needed to handle the specific data format of the simulation in a simple and efficient manner, and allow the user to apply a variety of visualization methods to the data. This implies the development and integration of various data conversion utility programs. Another purpose of this tool was to be able to use it to verify the correctness of the computational model if needed. It should be able to view and analyze the computational domain as well as the physical domain, and highlight areas that are obvious errors in the computation. Next, the tool needed to be able to easily generate graphics for the user to include in various papers and presentations. Since members of the team studying this water quality problem were located at various sites across the country, the tool needed to support various methods of collaboration. Lastly, and most importantly, all of this functionality needed to be available to the user in an easily understood interface.

Numerical Domain

The numerical domain used in this simulation consists of 10196 grid cells. The computational model outputs more than 100 scalar variables, and more than 10 vector variables. A routine run of the computation covers a whole calendar year generating 365 timesteps, and larger runs have also been made projecting out 10 and 20 years into the future.

The cells for the simulation are hexahedral in shape and are non-uniform in the x and y dimensions but all cells are uniformly spaced in depth. The grid itself is unstructured in

[1] Research Programmer, *rstein@ncsa.uiuc.edu*
[2] Research Scientist, *ashih@ncsa.uiuc.edu*
[3] Associate Director, *baker@ncsa.uiuc.edu*
[4] Research Hydrologist, *cerco@homer.wes.army.mil*
[5] Research Scientist, *noel@tempest.wes.army.mil*

nature, however, the cells are arranged by water columns (i.e. neighbors along the z-axis have the same dimensions in x and y). This is to accommodate the water depth variation at each location. Coordinates of the grid cell vertices are stored as latitudes and longitudes with the depth in meters. The coordinate system was converted into a Cartesian system in meters for the convenience of visualization. Dimensions in X and Y are relatively larger than the one in the Z (depth) direction. It thus makes the domain a very thin plate. In order to visualize the data within this domain more meaningfully, allowing the user to control the scaling in each direction was implemented.

Scalar constituent values are stored at the cell centers. In order to support visualization methods that require a continuous scalar field, such as isosurfacing, these values were also interpolated to the vertices of each cell. Vector constituent fields are supplied from the model, but their values reflect face-centered fluxes entering and exiting the cells. This is complicated by the fact that entering and exiting faces of a particular cell do not necessarily have the same geometry. In order to extract a meaningful vector field on which visualization methods could be applied, the first step is to find the surface normal of each face. This value is then used to decompose each face vector into its components and sum all components across the cell to obtain a correct cell-centered vector value. (See Figure 1.) In addition, each face-centered value was also normalized by the area of the cell. Since the vector field was based on the flux through an individual cell's faces, cells with smaller surface area had vectors with very small magnitude relative to the larger cells, and were therefore difficult to see. Normalizing each vector by the area of the cell allows the scientist to quickly examine the vector field to ensure the correctness of the model.

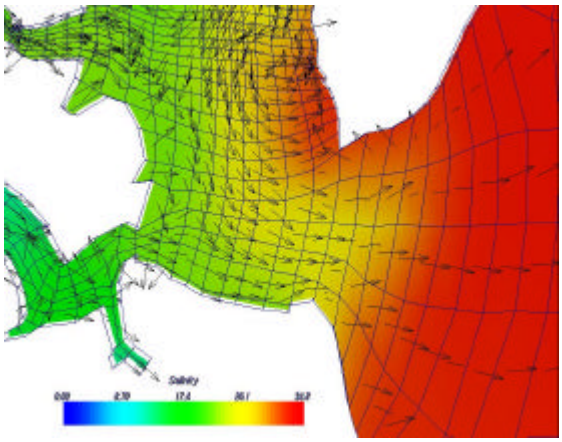


Figure 1. Cell Centered Vector Field in Physical Domain

To facilitate visual detection of anomalies in the computational model, the user is allowed to toggle the display between computational and physical domains. The display of the computational domain is simply a grid with the same structure where each cell of the grid is of uniform shape and size. This is desirable since many of the domain cells in the rivers of the bay are very tiny and difficult to see in the physical domain. In the computational domain all cells are the same size such that errors, even in small river cells, are easy to detect (Figure 2).

Visualization

We chose to use VTK (the Visualization Toolkit) for developing the visual representations for the application. VTK is an open-source, freely available software system for computer graphics, image processing, and visualization. It supports hundreds of algorithms in visualization and image processing fields.

To facilitate the development of a dynamic and flexible visualization application, a general library API called the Visualization Generator (VisGen) was developed. The Visualization Generator is a high level component for the development of 3D graphics applications. The VisGen library sits on top of VTK allowing the user to dynamically build VTK graphics pipelines for the production of interactive 3D scientific visualizations. The design of the VisGen is such that the implementation details of VTK are hidden from the novice user, but are available to experienced users for integration into complex applications and working environments.

The API of the VisGen is designed around a message-passing scheme where messages pass from the application layer to the VisGen. As a result of these messages, the VisGen produces a set of graphical objects that are available to the application for rendering. Since these messages are simple strings, the API makes it very easy to use applications in a collaborative fashion where application states are shared between multiple users. In addition, this type of messaging also easily supports rapid prototyping of new data, remote visualization, and easy integration with components of the user interface.

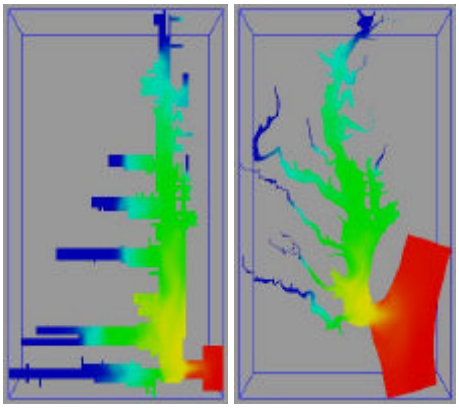


Figure 2. Computation vs. Physical Domains

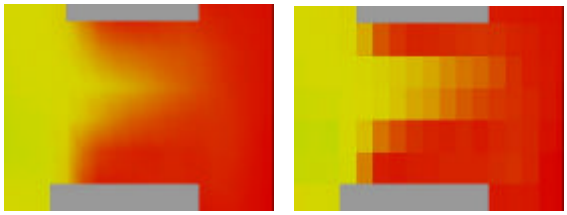


Figure 3. Cell-Centered vs. Vertex-Centered Slicing

The application supports a variety of visualization methods applying to scalar and vector data. These methods include slicing planes, isosurfacing, a representation of the external shell of the domain, and vector fields (Figures 8, 11).

The user is allowed to view the results of any of these methods in both the physical and computational domains.

In addition, depending on whether the user specifies cell-centered or vertex-centered data as input to a particular method, an appropriate representation is chosen if possible. (See Figure 3.)

Due to the actual dimensions of the domain (the main stem of the bay is about 300 km long and 8-48 km wide) the computational grids are relatively coarse. Consequently, the grid sizes may exceed the actual scales of the flow features as each cell face can be as big as more than 100 square kilometers. Therefore, the use of streamlines for visualizing the vector data may not always return meaningful results. The user can also specify a variety of color maps to apply to the representations allowing them more freedom in the final appearance of the image. In addition, there exists the ability to do custom titling, and color bars denoting the mapping of data to screen color are automatically positioned on the screen as each new dataset is added. For example, Figure 9 shows 4 variables, each with their own color legend and Figure 10 is a close-up view near the mouth of the bay.

We also experimented with building some functionality directly in OpenGL. Figure 4 shows an initial version of a tool to display the transport flux data. Though the tool provided the functionality of visualizing the vector data, the coding itself was far more complicated and laborious compared to the similar implementation using VTK.

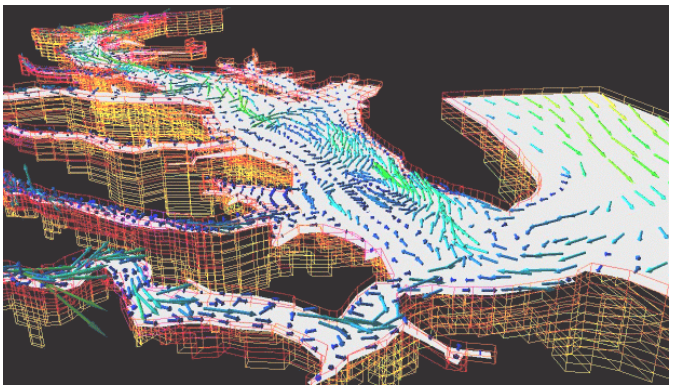


Figure 4. Initial prototype of vector data visualization implemented using OpenGL

Graphical User Interface (GUI)

As mentioned above, another important aspect in the design of this tool was that of the user interface. We based the current version of this interface on a library called the Fast Light Toolkit (FLTK) (Figure 6). FLTK provides a very comprehensive collection of widget classes, such as buttons and sliders, which allows users to build the GUI quickly and easily. It provides a visual programming environment called FLUID. This makes it very easy to design the GUI visually, and to integrate effectively with C++ code.

Unfortunately, there exists no standard way to interface the graphics window of a VTK based application with FLTK. A subclass of one of the FLTK windowing classes was therefore developed. It allows developers to include the VTK graphics, and track the user's mouse and button events in the FLTK GUI windows. This allows a developer to take the full advantage of both packages (Figure 7).

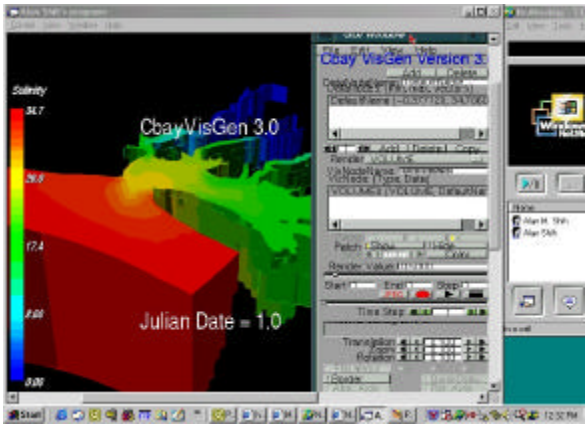
Because of the high level object-oriented nature of the VisGen library that we developed, the user can easily and intuitively add and remove different scalar and vector data fields for analysis. We call these DataNodes. Once a DataNode exists, the user can then apply any of the visualization methods available to this data. This generates a particular representation depending on the method, which is called a VisNode. The user is free to construct any combination of VisNodes and DataNodes to compose the image on the screen and can name each VisNode and DataNode that might make sense.

To support the user's need for graphics to use in publications and presentations, we included a variety of ways of exporting the current graphics window. First, and most simply, we provided the ability to save the current screen as a JPEG compressed image. This works nicely so that saved images can be put directly on the web without further conversion. The tool also has the ability to export a file to a high quality renderer called RenderMan. This software can give high quality, production level renderings of these images.

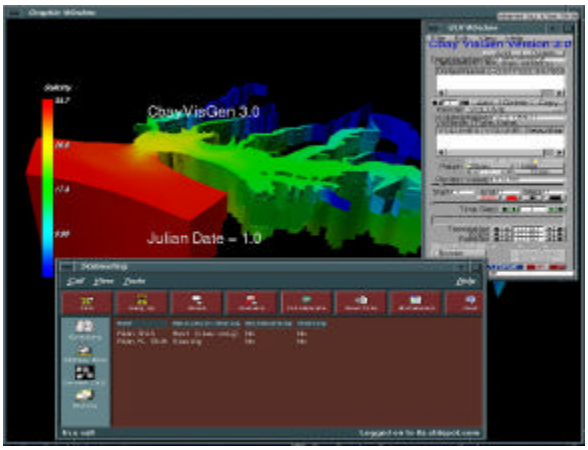
Of particular interest to the user was the ability to capture animations. This was made more difficult by the large number of timesteps in the computation, coupled with the large number of possible scalar and vector fields. Our tool incorporates this ability to play through animations and to capture QuickTime movies directly from the screen, for long numbers of timesteps and large resolutions. We have used the tool to capture animations at a resolution of greater than 1280 x 1024 with more than 365 timesteps. Lastly, the ability for the user to capture the current visualization as a VRML file capable of being viewed through a standard web browser are provided. This gives the user the ability to actually rotate the model of the bay and get a better feel for the spatial relationships.

Collaboration

The core members of the research team are located in multiple buildings at their site and they collaborate with members who reside in another state. We experimented with sharing the visualization sessions across the network using existing conference software. On a notebook computer running Windows 98, NetMeeting was used while SGIMeeting was used for an SGI Octane running IRIX 6.5. This visualization tool can be executed on the SGI and is shared between the PCs or SGI workstations using existing software. Since conferencing software provide many salient features such as control authorization, which allows different participant to take control of the visualization tool, collaborated visualization can be achieved seamlessly and efficiently. Figure 5 demonstrated the synchronized collaboration of the visualization tool.



(a) Windows 98 Environment using NetMeeting Software



(b) SGI IRIX 6.5 Environment using SGIMeeting Software

Figure 5. Collaboration between Windows PC and SGI Workstations

Summary

In conclusion, we have developed a comprehensive scientific visualization tool to handle the understanding and analysis of water quality in the Chesapeake Bay. Some problematic issues in efficiently handling and processing the data format from the computational model was resolved through this work, and a sophisticated system for dynamically generating visualizations of the data have been implemented. In addition, the development of the VisGen library allows for high-level, flexible control of the VTK graphics pipeline. Coupled with an easy-to-use interface to the application, this allows the user a lot of control over the graphical representation of the data. Once the user has a representation he/she is pleased with, a wide variety of options are provided for how this can be used in presentation, or for sharing with remote colleagues.

Acknowledgement

This work was carried out as part of the Programming Environment and Training component of the DoD High-Performance Computing Modernization Program.

References

1. Homepage for *The Fast Light Toolkit (FLTK)* http://www.fltk.org
2. Homepage for *The Visualization Toolkit (VTK)* http://www.kitware.com/vtk.html
3. William J. Schroeder, Kenneth M. Martin, and William E. Lorenson. "The Design and Implementation of an Object-Oriented Toolkit for 3D Graphics and Visualization," In *IEEE Visualization*, pp. 93-100, 1996.
4. William J. Schroeder, Kenneth M. Martin, and William E. Lorenson. *The Visualization Toolkit.* Prentice Hall PTR, 1996.
5. Carl Cerco, and Thomas Cole. "Three-Dimensional Eutrophication Model of Chesapeake Bay," In *Journal of Environmental Engineering*, pp. 1006-1025, 1993.
6. Adam Forgang, Bernd Hamann, and C. F. Cerco, "Visualization of water quality data for the Chesapeake Bay," In *IEEE Visualization*, pp. 417-420, 1996.

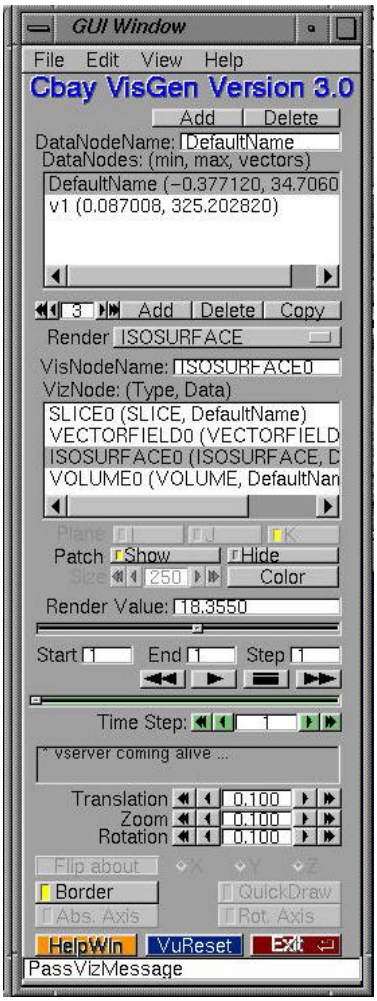


Figure 6. Graphical User Interface using FLTK Toolkit

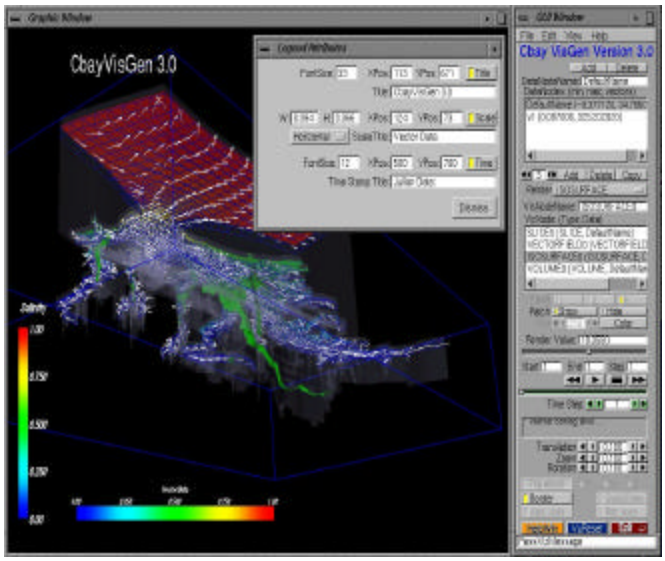


Figure 7. Binding Between FLTK and VTK allows the Development of a Comprehensive Package

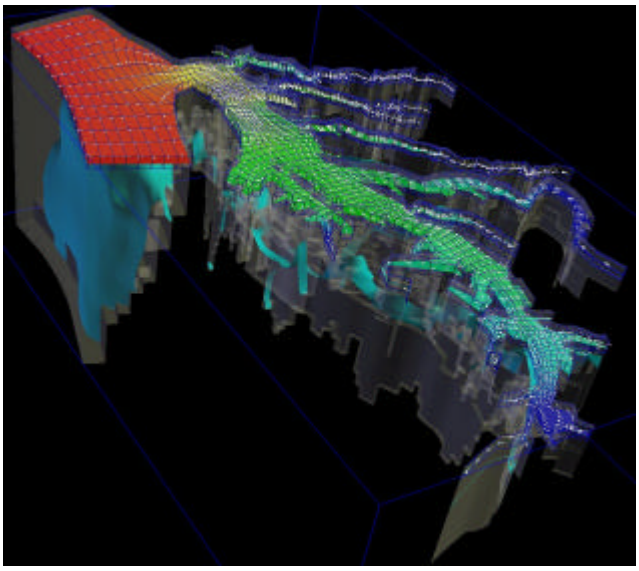


Figure 8. Combining a Slice Place of Salinity with Flow Vectors and an Isosurface of Dissolved Oxygen

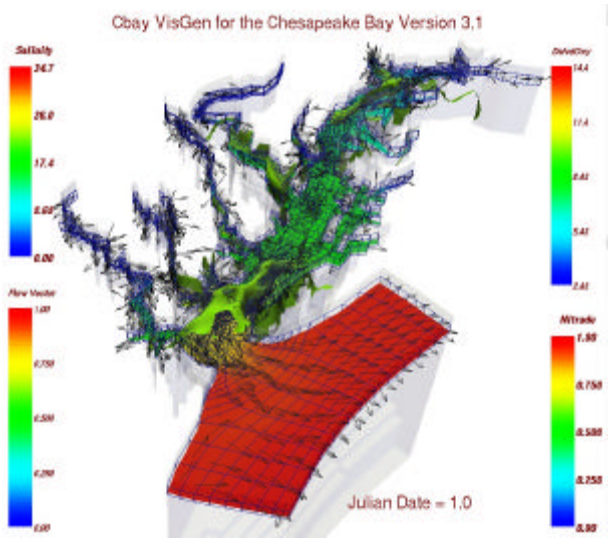


Figure 9. Complex Visual Composition of the Domain

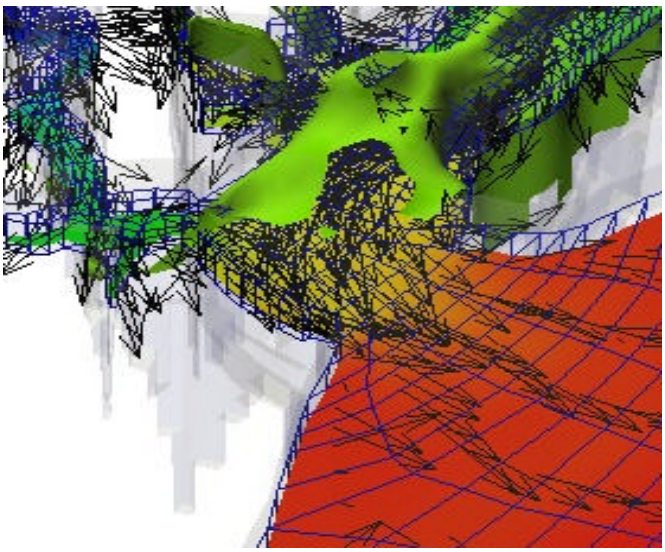


Figure 10. Close-up View near the Bay Mouth

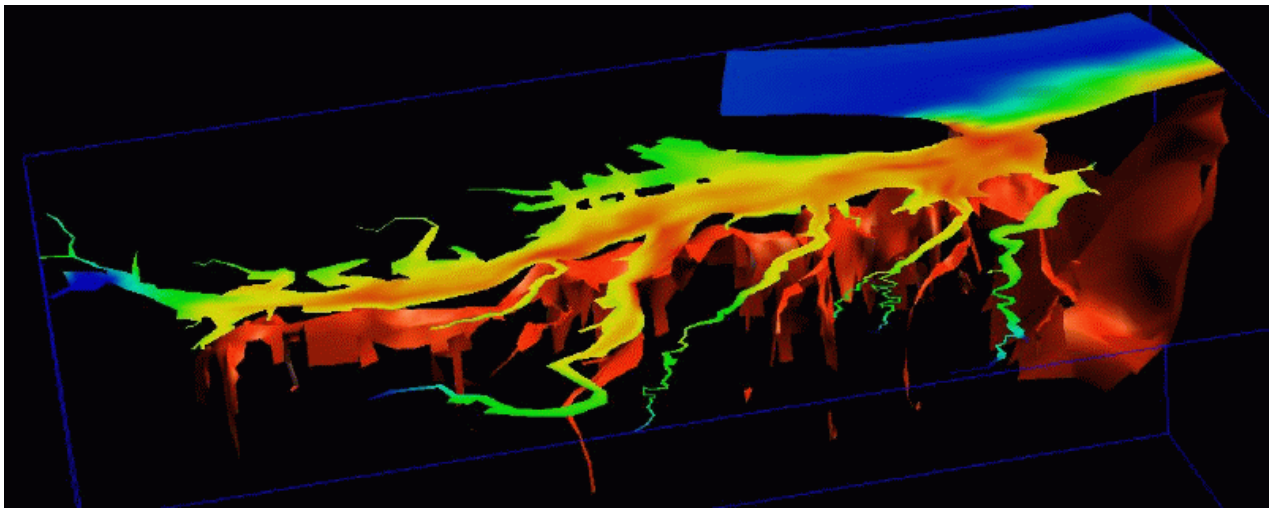


Figure 11. Slice and Isosurface Presentation